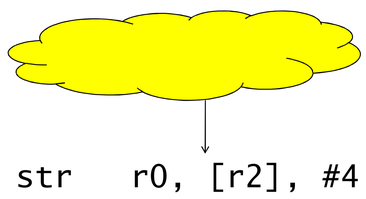
Comparing Program Logics for Reasoning about Safety Properties

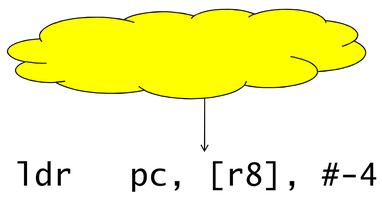
Lu Zhao John Regehr
School of Computing, University of Utah
{luzhao, regehr}@cs.utah.edu

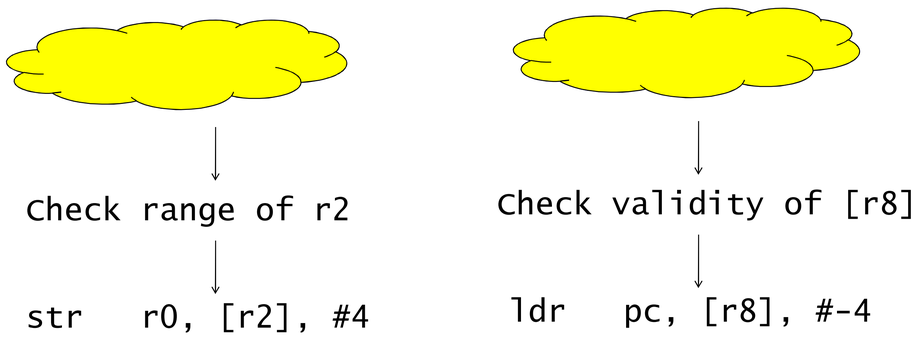
Background



Where is r2?

Where does pc go?





To prove memory writes and control transfers do not interfere with other programs in embedded systems.

Need a program logic with properties:

- Scaling to thousands of lines of machine code
- Capable of reasoning code with arbitrary control flow transfers
- Handling infinite loops

Hoare Logic with Separation Conjunction

Theorem is:

```
├ PROG_SPEC (Precondition)
        {Code}
        (Postcondition * Property)
```

✘ Almost impossible to write a sensible Hoare triple for thousands of lines of code.

✘ Extremely difficult to handle indirect control flow transfers.

✘ Unable to handle infinite loops.

Hoare Logic with Label Predicate

Theorem is:

```
├ PROG_SPEC ((Precondition UNION {(l, p)})
        {Code}
        ((Postcondition UNION {(k, q)} * Property)
```

✓ Enable flexible composition of specifications by matching labels.

✘ Composition may become intractable.

✘ Fail to handle infinite loops.

Hoare Logic and Condition Verification

The specification of a basic block is:

```
├ SAFE_BBL (Precondition)
        {Code}
        (Postcondition * Property)
```

The specification of a program is:

```
├ SAFE_PROGRAM guard prog predecessor =
  ∀ bbl ∈ prog. SAFE_BBL bbl ∧
    ∀ pre ∈ (predecessor bbl).
      ∀ branch ∈ (post_condition_of pre).
        branch jumps_to bbl ⇒ guard bbl
```

✓ Enable modular reasoning in the unit of basic blocks.

✓ Handle arbitrary control flow transfers.

✓ Handle infinite loops.

References

[1] Magnus O. Myreen, Anthony C. J. Fox and Michael J. C. Gordon. Hoare Logic for ARM Machine Code. In *Intl. Symposium on Fundamentals of Software Engineering*. 2007. Springer.

[2] Gang Tan. A Compositional Logic for Control Flow. In 7th Intl. Conf. on Verification, Model Checking and Abstract Interpretation. pp 80-94. 2006. Springer.

[3] Dachuan Yu, Nadeem A. Hamid and Zhong Shao. Building Certified Libraries for PCC: Dynamic Storage Allocation. In *Science of Computer Programming* 50 (2004) 101-127.